Domain: Application

DESCRIPTION

The Application Domain defines how service oriented applications are designed and developed. The Application Domain identifies open standards to facilitate rapid service oriented development, integration and implementation of new applications and business processes.

In addition to having well-defined SOA governance processes that identify and maintain high-quality service components, it is essential to have a standards-based service oriented development model (SODA), and development methodology that enables reuse of these components.

Developers must understand that the needs of service-oriented development are different from those of traditional and component-based applications. In service-oriented development, applications are developed from the start with Web Service standards based integration in mind. Design and development is done with multi-platform, inter-enterprise interoperability as a critical success factor.

SODA is the development paradigm for building applications supported by a Service Oriented Architecture. Using services as the primary unit of modularity requires a new approach that involves composing applications from sets of loosely integrated processes. This is an "assemble first" approach, rather than a "code first" approach used in traditional development. Code is necessary in a SODA world, but it is hidden behind the service interfaces. This means that consumers do not need to be as concerned with the structure or technology of the underlying program logic and platform. Many IT organizations assumed that building service-oriented applications was no different than building object-oriented or component-oriented systems for application development, but the following differences are critical:

- How testing services must be done
- How teams cooperate to deploy SOA systems
- What roles must exist and how to ensure that reuse occurs
- How to achieve effective service composition
- How to govern orchestrated services

The terms “application” and “service” are often used interchangeably. When used in this document, these terms are defined as follows:

Application: A software program composed of multiple “services” which addresses a specific set of functional requirements, e.g. a Professional Licensing application, and is comprised of a finite number of modules or sub routines which perform discrete functions, e.g. accept application, renew application, change address, etc.

Service: A stand alone software module which performs a specific discrete function, e.g. Change Address, is accessible through standards based internet protocols, and can be combined with other services to create a service oriented application.

STRATEGIC IMPORTANCE

Service oriented applications enable a high level of system integration, reuse of components and rapid deployment in response to changing government and business requirements. The specifications included in the Application Domain foster service oriented application creation, integration and deployment, through the use of Web Service standards. This includes the enabling of legacy applications for Web Services. Important benefits include enhanced programmer productivity, improved return on existing IT assets and resources and a better integrated customer experience.

RELATED TRENDS

In addition to the emergence of the service oriented development of applications model, the following related trends have significant relevance to SOA:

- Availability of open source solutions
- Emergence of web service standards based development environments
- Emergence of web service standards based deployment platforms
- Federated approaches to security and systems management
- Governments' desire to eliminate silos of information and services
- Homeland security motivated requirements for information integration
- Government initiatives to reduce healthcare costs
- Emergence of Web Service Registries as a mechanism for SOA governance
- Emergence of Enterprise Service Buses to provide a much more intelligent IP network

META projects that by the end of 2004, Web services technology and service-oriented architectures will promote widespread development of reusable application components. Through 2005/06, the complexity of component oriented, distributed applications will spur creation and adoption of pattern-driven development models. By 2007, model-driven approaches will represent a significant minority of application development efforts.

VISION

Service oriented application design and development has great potential for more swiftly delivering to the Commonwealth more interoperable, reusable, and less-expensive software. Developer productivity will be enhanced through the use of SODA, which reduces the need to write new code and automates portal development, user interface design and even rapid application deployment. Automation will reduce application complexity and will make it possible for agency business analysts to make system changes without requiring programmers to modify system code. The use of enterprise Shared Services will provide economies of scale which result in reduced cost and time to implementation. Instead of running on separate tracks, application development and integration will converge in this model.

ROADMAP

Current State

- Object oriented, component oriented, client server, and mainframe development models are employed at the Commonwealth
- Consistent development methodologies are rarely used
- Silos of information and services exist
- Agencies benefit from the availability of a limited number of Shared Services, such as:
  - *Messaging*: Enterprise Service Bus (See Integration Domain)
  - *Portal*: Content publishing and management (See Access & Delivery Domain)

Target State

- Service oriented development models (SODA) become widespread
- The Unified Process (UP) is adopted as a standard methodology for application development
- Legacy applications are Web Service enabled
- Silos of information and services are greatly reduced
- Agencies benefit from the availability of additional SOA Shared Services, such as:
  - *Security*: XML Gateways, Identity Management (see Security Domain)
  - *Discovery*: Web Services Registry (see Integration Domain)

BOUNDARY

The ETRM Application Domain addresses specifications and best practices for service oriented application design, development and deployment:

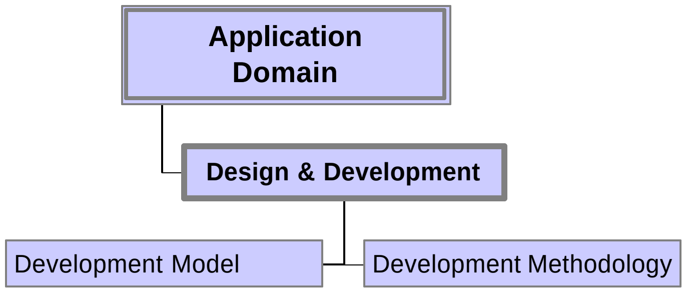
- Integration specifications and standards are defined in the ETRM Integration Domain.
- Security specifications and standards are defined in the ETRM Security Domain.

RELATED POLICIES

- Enterprise Open Standards Policy
- Enterprise IT Acquisition Policy
- SOA Governance (TBD)

ASSOCIATED DISCIPLINES

- Design & Development



Domain: Application

Discipline: Design & Development

DESCRIPTION

Service oriented development for applications (SODA) offers a strategy for eliminating the time and expense associated with traditional systems integration. This model helps makes design, development and integration process convergence a reality for the Commonwealth. SODA, based on open standards, lets developers become systems integrators working on a standardized platform (ISE) with a systematic methodology for building composite SOA applications. This service oriented model is best implemented using:

- An Integrated Design/Development Environment
- A Standard Development Methodology

This approach promotes a common skill set within agencies; one that uses standards, as well as graphical models, for creating exposable business interfaces that abstract system implementation details.

To maximize ROI, all agency software development projects should consider leveraging enterprise Shared Services as they become available:

- *Messaging*: Enterprise Service Bus (See ETRM Integration Domain)
- *Portal*: Content publishing and management (See ETRM Access & Delivery Domain)
- *Security*: Identity Management (See ETRM Security Domain)
- *Publish/Discover*: Web Services Registry (See ETRM Integration Domain)

Designing for application integration is the responsibility of SOA developers because the integration process should be planned from the inception of the system concepts. Focus on process is also emphasized in a SODA environment, as services are defined in a process model. This model describes the composition of services into a usable application (system process representation). It is composed of many micro models that represent the underlying business processes as services. Web services are made more adaptable through this model and through the subsequent use of rules, orchestration languages and parameterization. All of this is then controlled through the use of a registry/repository of components and services that is searchable. This repository represents the foundry of processes and may be populated from legacy "wrapped services," packaged application processes, data and unstructured content repositories, and newly created Web services.

RELEVANT STANDARDS ORGANIZATIONS

- **OASIS** – Organization for advancement of structured information standards is a not-for-profit, international consortium that drives the development, convergence and adoption of e-business standards. Members themselves set the OASIS technical agenda, using a lightweight, open process expressly designed to promote industry consensus and unite disparate efforts. OASIS produces worldwide standards for security, Web services, conformance, business transactions, supply chain, public sector, and interoperability within and between marketplaces. More information about OASIS can be found at http://www.oasis-open.org.

- **WS-Interoperability –** The Web Services Interoperability Organization is an open industry effort chartered to promote Web Services interoperability across platforms, applications, and programming languages. The organization brings together a diverse community of Web services leaders to respond to customer needs by providing guidance, recommended practices, and supporting resources for developing interoperable Web services, e.g. UDDI and WSDL. The organization's deliverables are targeted at proving resources for any Web services developer to create interoperable Web services, and verify that their results are compliant with both industry standards and WS-I recommended guidelines. More information about WS-I can be found at http://www.ws-i.org.
- **OMG (Object Management Group)** - The Object Management Group (OMG) is an open membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise applications. OMG membership includes virtually every large company in the computer industry, and hundreds of smaller ones. OMG is responsible for developing specifications for the Universal Modeling Language (UML). For additional information go to http://www.omg.org/gettingstarted/gettingstartedindex.htm

STAKEHOLDERS/ROLES

Designers, developers and implementers of Commonwealth web applications; enterprise application and data architects; business service architects and analysts; software developers and development service providers; data center infrastructure and operations planning.

ROADMAP

Currently, service oriented development techniques are not widely utilized. Application development tasks and application integration tasks are handled separately and often sequentially. There is no uniform development model or methodology. In the target state, service oriented development becomes the norm. Integration starts with the initial application design and a consistent development methodology is used throughout the enterprise.

ENTERPRISE TECHNOLOGY SOLUTION

Not applicable

ASSOCIATED TECHNOLOGY AREAS

- Development Model
- Development Methodology

Domain: Application
Discipline: Design & Development
Technology Area: Development Model

DESCRIPTION

Component and Object-Oriented development models are relatively well known in the enterprise. Service Oriented Development of Applications (SODA) is the next evolutionary phase in development models, building on the principles of these prior models, and additionally unifying the functions of application development and systems integration through the use of Web Service standards. Following this SODA model, using Component Oriented design and SOA Shared Services, yields other important benefits for the Commonwealth:

- Business logic is isolated in a separate layer that can exist well beyond the lifetime of any system using it, substantially increasing ROI and encouraging component reuse
- Use of a Portal Shared Service provides users with a single point of access for all Commonwealth services. In addition, a consistent presentation layer greatly enhances the government application user experience.
- Consistent service quality is derived from common standards and policies
- Reusable services supply basic building blocks that speed the development of new applications
- Run time service reuse only requires finding a service in the Registry and binding to it. This eliminates interoperability problems that typically impede code reuse, such as compiler versions, platforms and programming languages.
- Service location transparency, where applications look up services in a shared Registry, improves code mobility because services can be moved to different machines, or to external providers
- Support for multiple client types (on demand) with the same service, through layered application design, shared policies, and ESB Transformation Services, reduces development and maintenance costs while providing a more flexible user experience
- Use of a shared Enterprise Service Bus (ESB) provides an assured delivery mechanism that eliminates the need for developers to code for potential network failures
- Use of Security Shared Services eliminates the need for each application to perform authentications and maintain identity repositories

SODA designs include the following characteristics:

- Service assembly and orchestration
- Service registries and repositories
- Service publish, search, discovery, bind and invoke
- Standardized/published external interface specifications (XML schemas)

- Late-late binding (allows programs to dynamically select which services are needed at runtime)
- Federated identity and systems management
- Any to any transformation services
- Loosely coupled modules (message-oriented)

Services-oriented development subsumes object and component-oriented development because these lower-level concepts support the underlying mechanisms for building and deploying systems. Agencies can (and should) use objects or components to build services. Service Oriented Development of Application environments will continue to evolve and mature dramatically over the next few years, to fully support the elements of SODA that will be required to deliver applications based on service-oriented architectures. Developers should begin to look for the support of SODA's application development traits (modeling, orchestration etc.) within tools, as vendors incorporate them into products.

The successful creation of service-oriented applications requires:

- Design for extensibility and reuse
- Deciding which new functionality should be exposed as a service
- Loosely coupling services to support broad interoperability when requirements change
- Designing appropriate modularity and granularity of services

TECHNOLOGY SPECIFICATION: INTEROPERABILITY BASIC PROFILE

Description – The ETRM uses the WS-Interoperability Basic Profile as a baseline for the minimum SOA standards compliance required to insure interoperability across agencies and vendor platforms, e.g. J2EE and .NET. The Web Services Interoperability Organization (WS-I) is an open industry effort chartered to promote SOA Services interoperability across platforms, applications, and programming languages. The organization's deliverables include the WS-I Basic Profile, which allows developers to create interoperable Business Services, and verify that their results are compliant with both industry standards and WS-I recommended guidelines.

Standards & Specifications:

- WS-I Basic Profile 1.0 - All service oriented applications must be developed to support the WS-I Basic Profile 1.0 which includes:
  - XML 1.0 (Second Edition)
  - XML Schema Part 1: Structures
  - XML Schema Part 2: Data types
  - SOAP 1.1
  - WSDL 1.1
  - UDDI 2.0
  - RFC2246: The Transport Layer Security Protocol Version 1.0
  - RFC2459: Internet X.509 Public Key Infrastructure Certificate

- RFC2616: HyperText Transfer Protocol 1.1
- Secure Sockets Layer Version 3.0
- RFC2965: HTTP State Management Mechanism

For security specifications, please refer to the Security Domain.

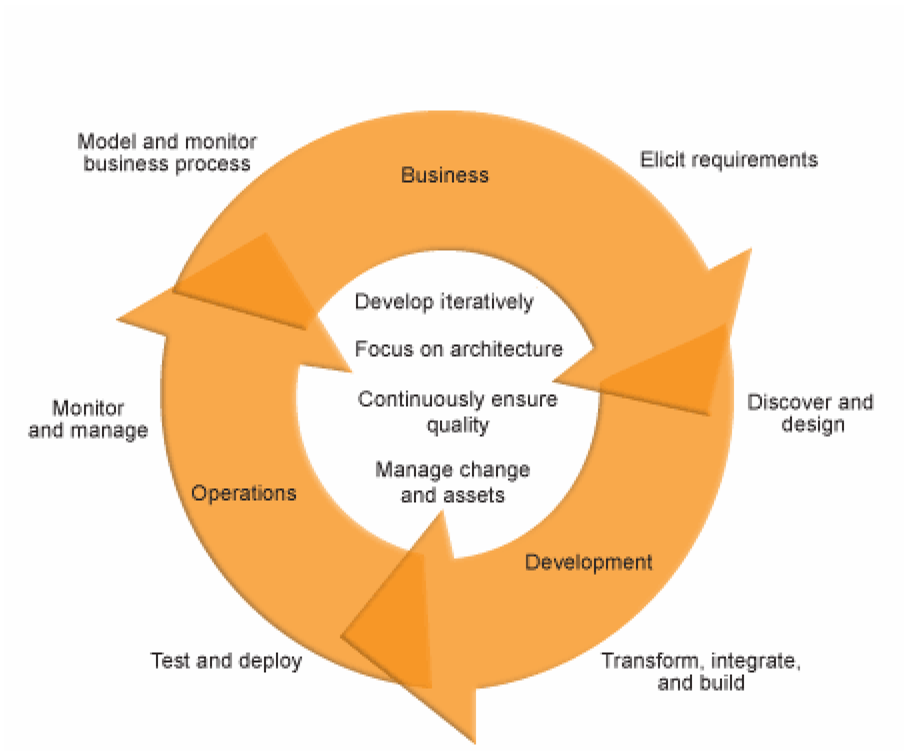
Migration Strategy – The ETRM will be updated to reflect any changes to the WS-I Basic Profile version to be implemented.

Domain: Application
Discipline: Design & Development
Technology Area: Development Methodology

DESCRIPTION

The diagram below describes the flow and processes that are typically found in a "best practices" software development methodology. Development methodologies are independent of the underlying development model, e.g. SODA. The outside of the circle shows the typical steps involved in software development. The center of the circle shows four principles or *imperatives* for a successful software project. These imperatives include:

- Develop iteratively -- don't try to develop the application in one pass.
- Focus on architecture -- use component models which you can reuse and apply in a service-oriented architecture (SOA).
- Continuously ensure quality -- test each design iteration to make sure quality is improving.
- Manage change and assets -- use software configuration and project management tools to control release levels, project requirements, and schedule integrity.



Source: IBM Corp.

TECHNOLOGY SPECIFICATION: UNIFIED PROCESS (UP)

Description – The UP (Unified Process) is an iterative software development methodology. Based on UML, UP organizes the development of software into four phases, each consisting of one or more executable iterations of the software at that stage of development.

- Inception - In this stage, the project's business case is stated and the team decides if the project is worth doing or if it is even possible. It is important to the process to first formulate the scope of the project and also determine what resources will be needed.
- Elaboration - In this stage, the developers take a closer look at the project to determine its architecture foundation and to evaluate the architecture in relation to the project. This stage is important to the UP because it is here that developers analyze the risks associated with changing the scope of the project or adding new technologies along the way.
- Construction - In this stage, the development of the project is completed. The application design is finished and the source code is written. It is in this stage that the software is tested to determine if the project has met its goal laid out in the inception phase.
- Transition - In this stage, any fine-tuning is performed. Any final adjustments can be based on user feedback, usability or installation issues.

Guidelines – Agencies should utilize a subset of UP, as appropriate for the project scope. There are a number of tools and products available designed to facilitate UP implementation. One of the more popular versions of UP is the Rational Unified Process (RUP)®.

Standards & Specifications –

- Agencies must use UP as a standard development methodology.
  Refer to: http://en.wikipedia.org/wiki/Rational_Unified_Process

Migration Strategy – Agencies should migrate to the Unified Process as a consistent methodology for application and/or service development.